난독화 알고리즘을 이용한 자바 역컴파일 방지기법에 관한 연구

안화수
고려대학교 컴퓨터정보통신대학원
e-mail : guardian23@korea.ac.kr

A Study on the Java Decompilation-Preventive Method by Obfuscating Algorithm

Hwa-Su Ahn
Graduate School of Computer and Information Technology,
Korea University

요 약

자바 언어의 중요한 특징 중의 하나는 어떤 기계에서든지 실행될 수 있다는 점이다. 이러한 플랫폼에 대한 독립성은 자바 프로그램이 바이트 코드 형태로 배포되기 때문에 가능한 일이다. 바이트 코드는 특정 기계에 종속되지 않고 자바 가상 머신(Java Virtual Machine:JVM)를 지원하는 곳이면 어디에서든지 실행 가능하다. 그런데 바이트 코드로 번역된 코드에는 자바 소스 코드의 정보가 그대로 포함되어 있는데, 이로 인해서 바이트 코드에서 자바 소스코드로의 역컴파일(Decompilation)이 쉽게 이루어진다는 취약점이 있다. 본 논문에서는 자바 바이트 코드의 난독화 기법을 살펴보고, Code Encryption Algorithm을 이용해서 역컴파일 하기 어려운 형태로 만드는 기술인 코드 난독처리(Code Obfuscation) 기법을 제안하였다.

1. 서론

자바 언어는 객체 지향적이고, 플랫폼에 독립적으로 실행되는 프로그래밍 언어적인 특징을 가지고 있다. 이러한 점이 특정 운영체제에 종속되지 않고 광범위 하게 사용될 수 있다는 것이 자바 언어의 장점이 될 수도 있지만, 또 한편으로는 이러한 장점 때문에 보안상 심각한 문제를 야기 시킬 수 있다[1]. 즉 자바 언어로 작성된 소스를 컴파일 하게 되면, 중간단계로서 바이트 코드(Byte Code)를 생성하게 되고, 이렇게 생성된 바이트 코드는 자바 가상 머신(Java Virtual Machine)에 의해서 자바 파일을 실행시키게 해준다. 이러한 플랫폼 독립적인 자바 가상 머신 상에서의 바이트코드의 실행가능성은 역공학적(Reverse Engineering)인 관점에서 볼 때 자바 클래스 파일을 자바 소스 파일로 역컴파일 하는 것을 가능하게 할 수 있다는 가능성을 포함하고 있다[2][3][4].

따라서 본 논문에서는 자바 바이트 코드의 난독화 기법을 살펴보고, Code Encryption Algorithm을 이용해서 역컴파일 하기 어려운 형태로 만드는 기술인 코드 난독처리(Code Obfuscation) 기법을 제시한다.

2. 관련 연구

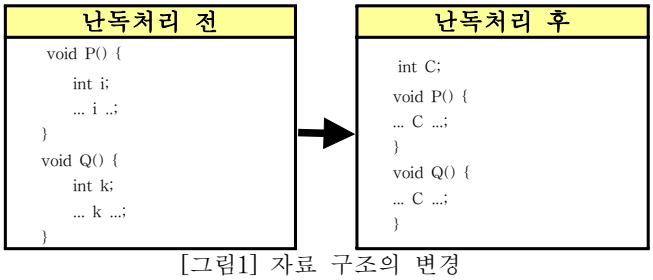
난독화는 프로그램의 구조를 복잡하게 하고 불필요한 부분을 추가함으로 프로그램의 가독성을 떨어뜨리는 것을 말한다. 물론 코드 난독화를 통해서 프로그램 성능에 지나친 영향을 주게 되면 곤란하기 때문에 난독화 정도를 조정할 필요도 있다. 그렇다면 현재 제안되고 있는 난독화 기법에 대해서 살펴보겠다[5].

2-1. 변수 이름의 변경

자바 소스 파일의 변수 이름이나 주석 등은 프로그램의 실행과는 상관이 없는 부분인데, 이들 정보를 변경하여 역컴파일을 어렵게 한다. 변수 이름은 프로그램을 읽기 쉽게 하는 중요한 요소가 되므로 이를 난해한 단어로 바꿈으로서 역컴파일로 생성된 소스코드를 읽기 어렵게 만든다.

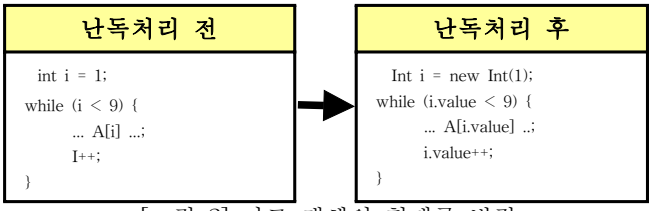
2-2. 프로그램의 자료구조를 변경

프로그램은 데이터를 조작함으로써 실행된다. 따라서 사용되는 자료구조를 변경하는 것은 난독처리의 중요한 부분이 된다. 다음은 데이터의 형태나 위치를 변경할 수 있는 예로서, 지역변수를 비지역변수로 바꾸는 것을 보여주고 있다.



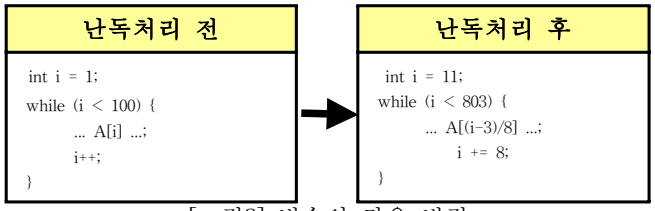
[그림1] 자료 구조의 변경

또한 자료 객체의 형태를 변경한 예로 다음과 같은 방법이 있다.



[그림 2] 자료 객체의 형태를 변경

변수 값을 변경한다. 예를 들어, 변수 i를 모든 위치에서 c1*i+ c2의 형태로 바꾸어도 프로그램의 실행에는 지장이 없다. 아래의 예는 c1이 8이고 c2가 3인 경우이다.



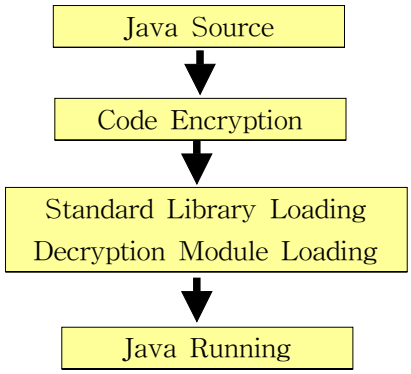
[그림3] 변수의 값을 변경

2-3. 프로그램의 제어를 변경

프로그램의 실행은 제어흐름이므로, 제어를 어지럽게 하면 프로그램은 그만큼 더 읽기 어려워지게 된다. 프로그램의 문장이 묶이는 단위를 조절한다. 예를 들어, 프로시저를 인라인(inline)화하는 방법이 있다. 이 방법에서 프로시저 호출 문장은 프로시저 자체로 대치된다. 아웃라인(outline)화하는 방법도 있다. 프로그램의 일부분을 떼어내어 독립된 하나의 프로시저로 구성하고 떼어낸 부분은 프로시저 호출 문장이 들어간다. 프로그램이 실행되는 순서를 바꾼다. 예를 들어, 1에서 10까지 증가하는 카운트는 10에서 1로 감소하는 카운트로 변경한다. 이 방법은 전통적인 컴파일러의 최적화의 기법으로 많이 사용되는 방법이다. 하나의 루프(loop)를 여러 개의 루프로 나누거나, 여러 개의 루프를 하나로 합치는 등의 테크닉도 컴파일러의 최적화 기법이기도 하면서 난독처리에서도 사용될 수 있다. 실행되지 않거나 실행되더라도 무의미한 결과를 산출하는 불필요한 코드를 일부러 집어넣어 읽기 어렵게 만든다. 실행되지 않는 코드를 데드코드(deadcode)라고 한다. 자바에 없는 코드를 삽입한다. goto문은 자바에는 없지만 자바 바이트 코드에는 존재한다. 따라서 goto가 존재하는 바이트 코드는 자바로 역컴파일 될 수가 없다.

3. Code Encryption Algorithm

Code Encryption Algorithm이란 자바 소스 파일의 특정 코드를 암호화하여 삽입하는 것을 말한다. 바이트 코드 파일 자체를 암호화 하는 것이 아니라 자바 소스 파일의 특정 코드를 암호화시켜 난독화를 시행 할 수 있다. 비록 역공학 소프트웨어를 이용하여 자바 소스 파일을 얻었다고 하더라도 소스 코드의 많은 부분이 암호화 되어 있기 때문에 악의를 가진 사용자는 정확한 소스를 파악 하기 힘들 것이다. 개발자는 필요에 따라서 소스 코드의 암호화율에 따라 난독화 정도를 조절할 수 있을 것이다. 다음은 자바 파일의 코드 암호화 및 실행 과정을 개략적으로 보여 주고 있다.



[그림4] code 암호화 과정 및 실행

4. 결론 및 향후 연구

자바 프로그램은 컴파일된 바이트 코드만 배포한다고 하여도 그로부터 소스코드를 쉽게 만들어 낼 수 있으므로 소프트웨어의 보호라는 측면에서 많은 취약점을 가지고 있다. 개발자가 많은 시간과 노력을 기울려 어렵게 개발한 소프트웨어를 악의의 사용자로부터 보호하려는 기법중의 하나가 난독처리 기법이다. 본 논문에서는 기존의 난독기법을 소개하였고, Code Encryption Algorithm을 이용한 난독 기법을 제안 하였다.

향후 연구 과제로서는 난독화를 위한 정.동적인 워터마크(watermark)의 사용에 대한 병행연구와 코드 암.복호화에 따른 오버헤드를 줄일 수 있는 연구도 병행되어야 할 것이다.

5. 참고문헌

[1] Neeran M. Karnik , Anand R. Tripathi, Design Issues in Mobile-Agent Programming Systems, IEEE Concurrency, v.6 n.3, p.52-61, July 1998

[2] G.Naumovich, N.Memon, Preventing Piracy, Reverse Engineering and Tampering, IEEE Press, 2003

[3] Jad - the fast Java Decompiler, http://www.kpdus.com/jad.html

[4] Mocha, the Java Decompiler, http://www.brouhaha.com/~eric/software/mocha/

[5] C. Collberg, C. Thomborson and D. Low, A Taxonomy of Obfuscating Transformations, Technical Report #148, Department of Computer Science, University of Auckland, New Zealand, 1997.